Stack

The Natural stack is a kind of "intermediate storage" in which you can store Natural commands, user-defined commands, and input data to be used by an INPUT statement.

This chapter covers the following topics:

- Use of Natural Stack
- Stack Processing
- Placing Data on the Stack
- Clearing the Stack

Use of Natural Stack

In the stack you can store a series of functions which are frequently executed one after the other, such as a series of logon commands.

The data/commands stored in the stack are "stacked" on top of one another. You can decide whether to put them on top or at the bottom of the stack. The data/command in the stack can only be processed in the order in which they are stacked, beginning from the top of the stack.

In a program, you may reference the system variable *DATA to determine the content of the stack (see the *System Variables* documentation for further information).

Stack Processing

The processing of the commands/data stored in the stack differs depending on the function being performed.

If a command is expected, that is, the NEXT prompt is about to be displayed, Natural first checks if a command is on the top of the stack. If there is, the NEXT prompt is suppressed and the command is read and deleted from the stack; the command is then executed as if it had been entered manually in response to the NEXT prompt.

If an INPUT statement containing input fields is being executed, Natural first checks if there are any input data on the top of the stack. If there are, these data are passed to the INPUT statement (in delimiter mode); the data read from the stack must be format-compatible with the variables in the INPUT statement; the data are then deleted from the stack. See also *Processing Data from the Natural Stack* in the INPUT statement description.

If an INPUT statement was executed using data from the stack, and this INPUT statement is re-executed via a REINPUT statement, the INPUT statement screen will be re-executed displaying the same data from the stack as when it was executed originally. With the REINPUT statement, no further data are read from the stack.

When a Natural program terminates normally, the stack is flushed beginning from the top until either a command is on the top of the stack or the stack is cleared. When a Natural program is terminated via the terminal command %% or with an error, the stack is cleared entirely.

Placing Data on the Stack

The following methods can be used to place data/commands on the stack:

- STACK Parameter
- STACK Statement
- FETCH and RUN Statements

STACK Parameter

The Natural profile parameter STACK may be used to place data/commands on the stack. The STACK parameter (described in the *Parameter Reference*) can be specified by the Natural administrator in the Natural parameter module at the installation of Natural; or you can specify it as a dynamic parameter when you invoke Natural.

When data/commands are to be placed on the stack via the STACK parameter, multiple commands must be separated from one another by a semicolon (;). If a command is to be passed within a sequence of data or command elements, it must be preceded by a semicolon.

Data for multiple INPUT statements must be separated from one another by a colon (:). Data that are to be read by a separate INPUT statement must be preceded by a colon. If a command is to be stacked which requires parameters, no colon is to be placed between the command and the parameters.

Semicolon and colon must not be used within the input data themselves as they will be interpreted as separation characters.

STACK Statement

The STACK statement can be used within a program to place data/commands in the stack. The data elements specified in one STACK statement will be used for one INPUT statement, which means that if data for multiple INPUT statements are to be placed on the stack, multiple STACK statements must be used.

Data may be placed on the stack either unformatted or formatted:

- If unformatted data are read from the stack, the data string is interpreted in delimiter mode and the characters specified with the session parameters IA (Input Assignment character) and ID (Input Delimiter character) are processed as control characters for keyword assignment and data separation.
- If formatted data are placed on the stack, each content of a field will be separated and passed to one input field in the corresponding INPUT statement. If the data to be placed on the stack contains delimiter, control or DBCS characters, it should be placed formatted on the stack to avoid unintentional interpretation of these characters.

See the *Statements* documentation for further information on the STACK statement.

FETCH and RUN Statements

The execution of a FETCH or RUN statement that contains parameters to be passed to the invoked program will result in these parameters being placed on top of the stack.

Clearing the Stack

The contents of the stack can be deleted with the RELEASE statement. See the *Statements* documentation for details on the RELEASE statement.

Note:
When a Natural program is terminated via the terminal command %% or with an error, the stack is cleared entirely.